ГОСУДАРСТВЕННЫЙ КОМИТЕТ
ПО ИСПОЛЬЗОВАНИЮ АТОМНОЙ ЭНЕРГИИ СССР

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

И Ф В Э 86-228
ОМВТ/ОП/ОУНК

Л.А.Егошин, П.Н.Казаков, Ю.В.Куянов, В.П.Сахаров

# РАСШИРЕНИЕ СРЕДСТВ ОПЕРАЦИОННОЙ СИСТЕМЫ ДЕМОС ДЛЯ ЗАДАЧ РЕАЛЬНОГО ВРЕМЕНИ

Серпухов 1986

УДК 234.384.6.

М-24

Аннотация

Егошин Л.А., Казаков П.Н., Куянов Ю.В., Сахаров В.П. Расширение средств операционной системы ДЕМОС для задач реального времени: Препринт ИФВЭ 86-228. - Серпухов, 1986. - 12 с., библиогр.: 4.

В работе представлен драйвер аппаратуры КАМАК для перспективной и широко распространенной портабельной операционной системы ДЕМОС. Приведены также результаты испытаний драйвера на мини-ЭВМ "Электроника 100/25". В приложении даются описание драйвера и конкретные рекомендации по его использованию.

Abstract

Egoshin L.A., Kazakov P.N., Kuyanov Yu.V., Sakharov V.P. An Operating System DEMOS Utilities Extention for On-Line Application: IHEP Preprint 86-228. - Serpukhov, 1986. p. 12, refs.: 4.

A CAMAC driver for advanced, portable and widely used operating system DEMOS is described. The results of testing the driver on minicomputer "Electronica 100/25" are also presented. The description of the driver and particular instructions for its application are given in Appendix.

# ВВЕДЕНИЕ

Изменения спектра задач реального времени и повышение требований к уровню сервиса пользовательского интерфейса в диалоговых системах сбора и обработки данных требуют расширения инструментального набора в известных операционных системах (ОС) таких, как RSX-11, RT-11, UNIX, ОС РВ, РАФОС, ДЕМОС.

Цель данной работы состояла в реализации программных средств для систем с аппаратурой в стандарте КАМАК в ОС ДЕМОС[1], которая входит в семейство ОС, совместимых с ОС UNIX[2]. Здесь приведены основные измеренные характеристики разработанного драйвера для широко применяемого крейт-контроллера СС-11 (СС-U2). Результаты измерений сравниваются с аналогичными характеристиками некоторых других ОС реального времени.

## 1. НЕКОТОРЫЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РАБОТЫ С АППАРАТУРОЙ КАМАК

Наиболее простым способом работы с аппаратурой КАМАК является прямая адресация к его модулям через адресное пространство, открываемое крейт-контроллером[3]. Однако возникающие при этом вопросы мультидоступа к аппаратуре КАМАК, т.е. возможность использования несколькими процессами одних и тех же сигналов LAM, модулей и крейтов, должны решаться пользователем самостоятельно. Альтернативой, позволяющей разрешить возникающие проблемы в многопользовательском режиме работы с аппаратурой КАМАК, является работа через драйвер крейт-контроллера. Создание драйвера в ОС ДЕМОС позволяет, используя развитые инструментальные средства, в значительной степени улучшить диалоговый интерфейс системы сбора и обработки данных в реальном масштабе времени.

## 2. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ И ОСОБЕННОСТИ РЕАЛИЗАЦИИ

Исходя из соображений унификации программного обеспечения аппаратуры КАМАК, программный интерфейс к драйверу учитывает подмножество рекомендаций ESONE/SR/01 (сентябрь 1978)/3/, включая блочные обмены через экстракоды к ядру системы. Драйвер в ОС ДЕМОС функционирует следующим образом: после прохождения через ядро системы экстракода типа ioctl, выданного из программы пользователя, управление процессом передается драйверу; после выполнения необходимых действий в целях защиты системы драйвер извлекает параметры и, если заказано ожидание сигнала LAM, реализует ожидание, а затем выполняет блочный обмен. Возвращаемым параметром в процесс пользователя является количество завершенных обменов.

Драйвер написан на языке Си (200 операторов) и на ассемблере (около 400 операторов). Объем занимаемой оперативной памяти - 2656 байтов. Наиболее важным параметром в системе реального времени является время на запись или чтение одного регистра, которое в этом случае составляет от 1,8 до 2,4 мс на один экстракод. Для сравнения в RSX-11M время, необходимое для чтения CSR ( регистра состояния КАМАК), составляет 3,2-4,0 мс. Также измерены полные затраты времени на прием одного события, имеющего следующую структуру:

- ожидание сигнала LAM;
- прерывание процессора;
- запуск процесса вторым проходом через диспетчер ОС ДЕМОС;
- адресное сканирование одного крейта;
- возврат в программу пользователя.

Полученные значения этого параметра составляли от 5,7 до 6,2 мс. В RSX-11M среднее значение аналогичного параметра около 6 мс.

## ЗАКЛЮЧЕНИЕ

Реализация в ОС ДЕМОС программных средств для задач реального времени позволяет широко использовать развитый программный инструментарий для диалоговых систем. Кроме того, благодаря портабельности ОС ДЕМОС при переходе на ЭВМ с другой архитектурой, модификации уже созданного программного обеспечения/4/ будут незначительными. Существенные преимущества для режима реального времени представляет возможность использования сателлитного процессора в сетевой организации ОС ДЕМОС.

Авторы считают своим долгом выразить признательность А.А.Лебедеву за стимулирующие обсуждения по постановке задачи, и С.В.Клименко за настойчивость по отношению к авторам, благодаря которой появилась эта работа.

## Литература

1. Операционная система ДЕМОС. Описание применения.–МПО–ЦПС, Калинин, 1985.
2. The Bell System technical journal, July-August 1978.
3. Бейлин М.В., Вьюшин О.В. и др. – Препринт ИЯФ 82–72, Новосибирск, 1982.
4. Вьюшин О.В., Храпкин П.Л. – Препринт ИЯФ 82–74, Новосибирск, 1982.

Рукопись поступила 3 декабря 1986 года.

## Приложение 1.

## Инструкция по включению драйвера в ядро системы

Сначала обязательно убедитесь, что ядро операционной системы ДЕМОС собирается нормальным образом, т.е. что у вас имеются в библиотеках откомпилированные модули ядра и драйверов.

Разыщите файлы "m.h", "l.s", "c.c", "makefile" и один из трех файлов "m40.c", "mch.c" или "smch.c" - в зависимости от типа генерации. Они обычно находятся в директории "conf".

В файле "m.h" указано, использует ли ваше ядро пространство данных, раздельное от пространства команд (для ЭВМ типа "Электроника-79"). Это параметр SEPID. Если он у вас определен, т.е. у вас ЭВМ "Электроника-79", то необходимо изменить первые две строки файла "cc2.s" драйвера КАМАК:

```
mfpd    =       106500          / mfpi for 11/40
mtpd    =       106600          / mtpi for 11/40
```

Теперь отредактируйте один из трех файлов - "m40.c", "mch.c" или "smch.c" (Какой конкретно - это вы можете прочитать в первых строчках файла "makefile", там указано). Необходимо найти определение переменной "nofault" в этом файле. Это строка типа

```
nofault:        0
```

Вставьте перед ней, пожалуйста, строку

```
.globl  nofault
```

Затем отредактируйте файлы "l.s" и "c.c", содержащие таблицы векторов прерываний и таблицы адресов вызова драйверов. Что касается "c.c", то вместо какого-либо неиспользуемого драйвера в таблице "cdev" вставьте строку переходов в драйвер:

```
ccopen, nulldev, nodev, nodev, ccioctl, nulldev, 0,
```

а перед таблицей опишите точки входа:

```
int ccopen(), ccioctl();
```

В файле "l.s" по адресу векторов прерываний в крейте КАМАК впишите

```
cctr1;  br7+0   / LAM 1
. . . . . .
cctr1;  br7+7   / LAM 8
```

а в конце файла добавьте

```
.globl  _ccint
cctr1:  jsr     r5,trap
```

```
mov     <адрес>,-(sp)
jsr     pc,_ccint
tst     (sp)+
ret     pc
```

где <адрес> - это адрес регистров крейта КАМАК (160000, 162000, 164000 или 166000). Для нескольких крейтов в файле "1.s" надо расписать вектора всех крейтов и добавить аналогичные подпрограммы в конце файла, заменяя имена cctr1 на cctr2 и т.п.

**Предупреждение**

Адреса регистров **обязаны** располагаться последовательно, по стандартным адресам, иначе не будет работать автоматический переход на следующий крейт при адресном сканировании.

Крейты имеют номера (c) в порядке возрастания приведенных адресов. Если какой-либо отсутствует, то этот номер не будет "откликаться".

Вместо редакции файлов "1.s" и "c.c" можно откорректировать текст программы "mkconfig". Эта программа строит файлы "1.s" и "c.c", задавая конфигурацию вашего ядра.

Поместите файл "cc.h" в директорию "h", файл "camac.h" в директорию "/usr/include/sys", а файлы "cc1.c" и "cc2.s" в директорию "dev". Откомпилируйте драйвер в директории "dev":

```
cd ../dev
cc -O -c cc1.c
as -o cc2.o cc2.s
ar r LIB2 cc1.o cc2.o
```

Наконец,можно собирать ядро:

```
make demos
```

Не забудьте проверить магическую границу в 24К для ЭВМ типа СМ-4 и "Электроника-79".

После сборки в директории "/dev" надо завести специальный файл для связи с драйвером, например:

```
/etc/mknod /dev/camac c X 0
```

где вместо X должен стоять номер строки из таблицы "cdev" файла "c.c", куда вы поместили строку переходов к драйверу (строки нумеруются с нуля).

Если вы знакомы с любым редактором ОС ДЕМОС, читали руководство системного программиста, и уже собирали ядро, то вся процедура не должна занимать у вас более 15-20 мин.

## Приложение 2.

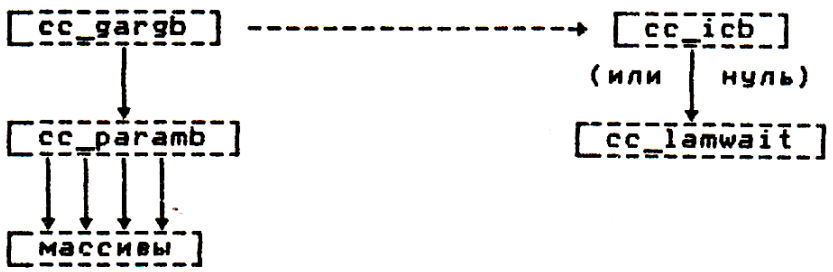
## Управляющие блоки и их взаимосвязь в операциях обмена данными

### Описание управляющих блоков

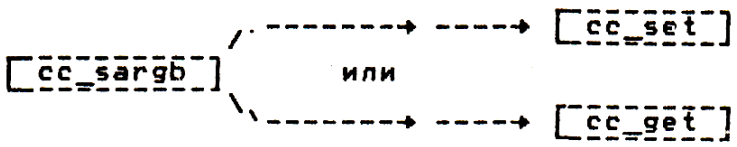
```
struct  cc_paramb                   /* Блок параметров                  */
{
        int     *cc_jfa;            /* адрес массива функций            */
        cc_ext  *cc_exta;           /* адрес массива CNA                */
        int     *cc_array;          /* адрес массива данных             */
        int     *cc_qa;             /* адрес массива ответов            */
};

struct  cc_icb                      /* Блок управления блочными         */
{                                   /*      операциями                  */
        int     cc_count;           /*   начальный     счетчик          */
        int     cc tally;           /* out param  - сколько сделано     */
        struct  cc_lamwait          *cc_lamid;      /* ид. LAM / 0      */
        int     cc_chan;
};

struct  cc_gargb                    /* Блок аргументов к вызову         */
{                                   /*           ioctl                  */
        struct  cc_paramb           *cc_pprm;       /* адрес paramb     */
        struct  cc_icb              *cc_picb;       /* адрес   icb      */
};

struct  cc_lamwait                 /* Блок управления ожиданием LAM     */
{
        struct  cc_lamwait          *cc_flink, *cc_blink;
        cc_ext  cc_glext;                   /* адрес CNA - крейт        */
        int     cc_gmask;                   /* маска LAM                */
        int     (*cc_fnc)();
};

struct  cc_sargb                   /* Блок аргументов к операциям       */
{                                  /* над управляющими рег. CSR/DMR     */
        cc_ext  *cc_sext;                   /* адрес CNA крейта         */
        union   {
                struct  cc_set  *cc_pset;       /* адрес cc_set         */
                struct  cc_get  *cc_pget;       /* адрес cc_get         */
```

```
        } cc_s;
};

struct  cc_set                  /* Блок данных установки или     */
{                               /*       сброса регистра CSR     */
        unsigned        cc_bic; /*   маска установки битов       */
        unsigned        cc_bis; /*   маска сброса битов CSR      */
};

struct  cc_get                  /* Блок данных из рег. CSR/DMR   */
{
        unsigned        cc csr; /*                CSR   out param */
        unsigned        cc dmr; /*                DMR   out param */
};
```

**Блочные операции обмена данными**




**Операции управления крейтом КАМАК**



**Приложение 3.**

## Запросы к драйверу КАМАК

В данном разделе описаны запросы к драйверу КАМАК на базе системного вызова ioctl. Все запросы сделаны таким образом, чтобы на их основе можно было написать пакет подпрограмм (п/п), реализующий стандарт ESONE/SR/01 на обмены с КАМАК.

В начале программы на языке Си необходимо поместить три следующих оператора:

```
 include        <sys/camac.h>
int     fd;
fd = open("/dev/camac",2);
```

Кроме того, необходимо создать структуры управляющих блоков. Для краткости будем считать их имена совпадающими с именами структур, но без префикса "cc_".

### 1. Операция опроса управляющих регистров CSR/DMR

```
sargb.cc_sext = &<адрес крейта>;
sargb.cc_s.cc_pget = &get;
ioctl(fd,SOPG,&sargb);
```

После операции в структуре get лежит информация из регистров CSR и DMR. На базе этого запроса реализуются п/п стандарта CTCC, CTLM, CTCI путем проверки соответствующих битов. Если крейт отключен, то системный вызов завершается с кодом ошибки.

### 2. Операция очистки и установки битов управляющего регистра CSR

```
sargb.cc_sext = &<адрес крейта>;
sargb.cc_s.cc_pset = &set;
set.cc_bic = <маска для сброса>;
set.cc_bis = <маска для установки>;
ioctl(fd,SOPS,&sargb);
```

Сначала по маске будут очищены биты регистра CSR, а затем по другой маске будет произведена установка битов

CSR. Этим запросом реализуются функции стандарта CCCZ, CCCC, CCCI. Если крейт отключен, то завершение системного вызова с кодом ошибки.

## 3. Блочный обмен с крейтами

Во всех операциях блочного обмена должна быть заполнена управляющая структура gargb:

```
gargb.cc_picb = &icb;
gargb.cc_pprm = &paramb;
```

Кроме того, заполняется структура icb. В ней заполняется начальный счетчик операций блочного обмена (число слов) icb.cc_count и поле icb.cc_lamid, в котором должен лежать либо нуль, либо адрес структуры lamwait, если необходимо ожидать возникновения сигнала LAM из какого-либо крейта до операции (см далее):

```
icb.cc_count = <число циклов обмена>;
icb.cc_lamid = &lamwait; /* или 0 */
```

В поле icb.cc_tally при завершении операции находится число выполненных циклов обмена. Если там нуль, то возможно, что у вас ошибка в управляющих структурах.

Если в операциях UBC, UBR, UBL или MAD задана функция управления КАМАК, то адрес массива данных задавать не обязательно.

Во всех запросах можно использовать при обмене 24 бита данных. Для этого надо ко второму аргументу ioctl добавить флаг DFLAG, например:

```
ioctl(fd,MAD|DFLAG,&gargb);
```

Тогда размерность массива данных надо удвоить. Сначала в нем расположено старшее слово, а за ним младшее, затем следующий элемент обмена и т.д.

### 3.1 MA, XMA

```
paramb.cc_ifa = &<массива функций>;
paramb.cc_exta = &<массива адресов CNA>;
paramb.cc_array = &<массива данных>;
paramb.cc_qa = &<массива ответов>;
```

```
ioctl(fd,MA,&gargb);
ioctl(fd,XMA,&gargb);
```

Запрос вызывает выполнение нескольких функций КАМАК. Число функций задается icb.cc_count и определяет используемое число элементов массивов. Выполняются функции по адресам, с использованием данных, а ответы кладутся в массив ответов. Если какая-либо функция есть функция управления, то соответствующий элемент массива данных не используется и по-другому читается или пишется. Операция MA возвращает либо 0 (нет Q), либо не 0 (есть ответ Q). Операция XMA возвращает два бита в каждом слове ответов:

```
0100000        -  ответ Q
0040000        -  ответ X
```

### 3.2 MAD

```
paramb.cc_ifa = &<функции>;
paramb.cc_exta = &<двух адресов CNA>;
paramb.cc_array = &<массива данных>;
ioctl(fd,MAD,&gargb);
```

Запрос выполняет последовательность функции КАМАК в режиме сканирования по адресам, т.е. указанную функцию применяют по первому адресу CNA. Если ответ Q положительный, то наращивают адрес CNA, или скачком идут к следующему модулю крейта с тем же словом массива данных. Операция продолжается до исчерпания массива данных (по icb.cc_count) или по достижению второг указанного адреса CNA. Операция автоматически продолжается через границы крейтов. Ответы Q не запоминаются.

### 3.3 UBC

```
paramb.cc_ifa = &<функции>;
paramb.cc_exta = &<адреса CNA>;
paramb.cc_array = &<массива данных>;
ioctl(fd,UBC,&gargb);
```

Запрос вызывает обмен массивом с КАМАК по адресу CNA, при помощи указанной функции. Обмен считается законченным при исчерпании счетчика icb.cc_count или при получении отрицательного ответа Q. Ответы не запоминаются.

### 3.4 UBL

```
paramb.cc_ifa = &<функции>;
```

```
paramb.cc_exta = &<адреса CNA>;
paramb.cc_array = &<массива данных>;
icb.cc_lamid = &lamwait;
lamwait.cc_gmask = <маска сигнала LAM>;
lamwait.cc_glext = <адрес CNA крейта>;
ioctl(fd,UBL,&gargb);
```

Обмен массивом с КАМАК по адресу CNA при помощи указанной функции. Используется прерывание процессора по LAM как сигнал к передаче очередного элемента массива. Обмен завершается по отрицательному ответу Q или исчерпанию счетчика. Ответы не запоминаются.

### 3.5 UBR

```
paramb.cc_ifa = &<функции>;
paramb.cc_exta = &<адреса CNA>;
paramb.cc_array = &<массива данных>;
ioctl(fd,UBR,&gargb);
```

Обмен массивом с КАМАК по адресу CNA при помощи указанной функции. Ответ Q используется как признак успешно завершенного обмена элементом массива. В противном случае обмен этим же элементом повторяется в цикле. В счетчике неуспешные циклы не считаются. Обмен завершается по исчерпанию счетчика или при завершении 4000-го неуспешного цикла подряд (нет модуля).

## 4. Особенности ожидания сигналов LAM

В случае использования блочных передач имеется возможность ожидать перед обменом появления сигнала LAM в аппаратуре КАМАК. Для этого в блоке icb должно быть установлено поле icb.cc_lamid. Оно должно содержать адрес блока lamwait, где задаются параметры ожидания. Если поле icb.cc_lamid равно нулю, то запрос выполняется немедленно.

В блоке lamwait заполняются поля:

```
lamwait.cc_glext = <адрес CNA крейта, где появится LAM>;
lamwait.cc_gmask = <битовая маска сигнала LAM>;
```

При помощи битовой маски можно ожидать только один сигнал LAM. Ожидание реализуется на уровне, доступном для сигналов, т.е. процесс сбрасывается сигналом с сис-

темного вызова точно так же, как при обмене с терминалом.

## 5. Адресация крейтов, модулей и регистров модулей

Адрес CNA везде представляет собой просто адрес регистра крейта на странице ввода-вывода. При обращении к упрвляющим регистрам (запросы SOPG и SOPG) используется только C из полного адреса.

В помощь программисту имеется тип данных cc_ext для представления адресов и несколько макроопределений в виде функций языка Си для формирования адресов:

```
cc_ext  cccna(c,n,a);    /* Сформировать адрес CNA          */
aint     c,n,a;

cc_ext  ccnexta(ext);    /* Сформировать адрес следующего */
cc_ext  ext;             /*       регистра                  */

cc_ext  ccnextn(ext);    /* Сформировать адрес следующего */
cc_ext  ext;             /*       модуля                    */

cc_ext  ccnextc(ext);    /* Сформировать адрес следующего */
cc_ext  ext;             /*       крейта                    */

cc_ext  ccchkext(ext);   /* Проверить адрес CNA на          */
cc_ext  ext;             /*       корректность              */
```

Функции ccnext и ccchkext возвращают нуль при выходе за допустимые пределы параметра ext. Наличие крейта не учитывается.

Л.А.Егошин и др.

Расширение средств операционной системы ДЕМОС для задач реального времени.

Редактор Н.В.Ежела. Технический редактор Л.П.Тимкина.

---

Подписано к печати 30.12.86. Т-25518. Формат 60x90/16.
Офсетная печать. Печ.л. 0,75. Уч.-изд.л. 0,83. Тираж 260.
Заказ 117. Индекс 3624. Цена 12 коп.

---

Институт физики высоких энергий, 142284, Серпухов Московской обл.

14 коп. Индекс 3624.

П Р Е П Р И Н Т 86-228, И Ф В Э, 1986.